IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

From Swarm Intelligence to Metaheuristics: Nature-Inspired Optimization Algorithms

Xin-She Yang, Middlesex University

Suash Deb, IT and Educational Consultant

Simon Fong, University of Macau

Xingshi He, Xi'an Polytechnic University

Yu-Xin Zhao, Harbin Engineering University

Nature has provided rich models for computational problem solving, including optimizations based on the swarm intelligence exhibited by fireflies, bats, and ants. These models can stimulate computer scientists to think nontraditionally in creating tools to address application design challenges.

Most if not all engineering tasks involve decisions about the product, service, and system design, which are related in some way to optimizing time and resources as well achieving balance between maximizing performance, profit, sustainability, quality, safety, and efficiency and minimizing cost, energy consumption, defects, and environmental impact. As the sidebar "Elements of an Optimization Problem" describes, many of these design problems have multiple objectives bound by highly complex constraints. The traditional approach of specializing design method to problem type does not fit well with complex, nonlinear problems, such as multicriteria engineering designs and multiple complex feature extraction in big data.

This lack of suitability has motivated interest in more novel optimization approaches. One emerging trend is to combine heuristic search with multiagent systems to solve real-world business and engineering problems.[1] Such a combination has accuracy, efficiency, and performance advantages over specialized methods. For example, it can be used to more efficiently and accurately optimize or tune parameters in artificial neural networks, which are essential to many AI tasks.

One class of novel optimization algorithms is based on swarm intelligence (SI). SI captures the idea that decision making among organisms in a community, such as ants and bees, uses local information and interactions with other agents and with their own environment, which in turn could be responsible for the rise of collective or social intelligence. One hypothesis is that complex interactions directly or indirectly contribute to the emergence of intelligence in highly developed biological species. The reasoning is that biological change results from the organism responding and adapting to alterations in its community and environment. Groups of

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

same-species organisms have successfully carried out specific tasks through collective or swarm intelligence.[2–4] When resources are scarce, different species cooperate and can even coevolve.

SI has inspired researchers to develop an array of ingenious methods for solving challenging problems such as optimization based on insect and bird behavior. These algorithms—or *metaheuristics*—have been applied to solve both machine learning and engineering problems. The application of these SI algorithms makes sense because, in many ways, an algorithm's iterative process strongly resembles a self-organized system's evolution.[5,6] Moreover, the noise and randomness in algorithms can promote diverse solutions and avoid the pitfall of being trapped in local optimal solutions. The result is that the selection mechanism more rapidly converges on global optimal solution. Just as in nature, iterative reorganization can lead to new structures of states, and such states can be considered as possible solutions to the problem at hand.

To better understand both SI and metaheuristics, we surveyed the literature to identify underlying ideas. The algorithms described are representative of the more than 140 nature-inspired algorithms and their variants.[7] However, this number is likely already lower than what actually exists, as research is producing new algorithms almost weekly. Thus, the algorithms we describe can be only illustrative—showing how inspiration from nature can be turned into effective problem solving. Our hope is that this survey is a springboard to novel thinking about methods and tools to address computational challenges.

Fundamental Concepts

Nature-inspired algorithms are based on SI, which in turn forms the foundation of metaheuristics.

Swarm intelligence

SI is a complex process and no one is yet certain what mechanisms are required to ensure the emergence of collective intelligence. However, extensive recent studies suggest that individual agents such as ants and bees follow simple rules, act on local information, and have no centralized control.[8] Such rule-based interactions can lead to the emergence of self-organization, resulting in structures and characteristics at a higher system level. Individuals in the system are not intelligent, but the overall system exhibits collective intelligence. Such emerging self-organization can explain key swarming behavior, whether in ants or people.

Certain conditions are necessary for systems to self-organize: feedback, stigmergy (when individual system parts intercommunicate indirectly by modifying their local environment), multiple interactions, memory, and environmental setting.[2,8] However, what role each condition plays and how seemingly self-organized structures can arise from these conditions are unclear, and different studies typically focus on one or a few of these influences.

Metaheuristics and monkeys

Most metaheuristic algorithms were developed from observing nature. Although quality solutions to a difficult optimization problem can be found in a heuristic way, there is no guarantee that the optimal solutions can be

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

found. Such heuristic approaches might work most of the time, but there is no guarantee that they will. In contrast, metaheuristics tend to carry out higher-level search in a vast design space and usually employ some memory to record historical best solutions in combination with some form of solution selection. One of the earliest metaheuristics was a quasi-random approach that simulated metals annealing[9] by assigning some probability linked to a system's energy consumption. When the temperature is high, the probability approaches one; when the temperature is low (nearly zero), the probability approaches zero. Thus, when the system cools, the algorithm is less likely to accept worse solutions, only better ones.

Because of their ability to conduct wider searches and record best solutions, metaheuristics tend to outperform simple heuristics.[7,10] The Infinite and Finite Monkey Theorems provides insights into the way these algorithms work.

Infinite Monkey Theorem. The Infinite Monkey Theorem assumes that, if in a group of monkeys, each is given a typewriter, the probability that they will produce any given text—for example, William Shakespeare's complete works—will almost surely be one if an infinite number of monkeys randomly type for an infinitely long time.[11,12] On a smaller scale, suppose the task was to reproduce the eight characters in "computer" from a random typing sequence of n characters on a keyboard of 101 keys. The probability that a consecutive eight-character random string will be "computer" is $p = (1/101)^8 \approx 9.23 \times 10^{-17}$, which is extremely small, but not zero.

Indeed, the central idea in this theorem is that the probability of reproducing any text given infinite monkeys is not zero. In fact, probability theory can be used to formally prove that, for an infinitely long sequence, the probability of reproducing any text (including this article) is almost surely one.[13]

Finite Monkey Theorem. However, it is impractical to assume that infinite resources are available, so a more reasonable perspective is the Finite Monkey Theorem: the probability is greater than zero that a finite number of monkeys typing randomly for a fixed time can reproduce some text. Transitioning that idea to people, a few billion people typing on keyboards daily has not yet produced another collected works of Shakespeare, but the probability that it will someday appear is greater than zero.

Defining differences. Metaheuristics and the monkey theorems have three crucial differences. First, monkeys type randomly without learning from or remembering their prior actions; metaheuristics try to learn from history to generate better solutions. Second, monkeys do not select what to type; metaheuristics tend to select the best or fittest solutions. That is, most use some form of randomization to enhance the search capability and thus increase the probability of finding truly optimal solutions. In theory, if such algorithms can be executed for a sufficiently long time with multiple runs, the best design solutions are likely to be found. However, an infinitely long search time is not possible, so most methods use thousands and even millions of design evaluations or iterations. Third, monkey typing is at most equivalent to a random search on a flat landscape of objectives, whereas metaheuristics use landscape information to guide the search process and generate new solutions—akin to hill climbing as the algorithm identifies better and better solutions.

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

Common Algorithmic Characteristics

Although algorithms can have different sources of inspiration and different mathematical equations, they share certain characteristics. For example, they all use a solution population, and some solutions in that population will be replaced as their probability of success decreases. Solutions are generated by modifying the population either locally or globally. The acceptance of new solutions is determined by their fitness and a higher fitness than existing ones will initiate a population update.

Thus, loosely speaking, all nature-inspired algorithms can be represented by the unified pseudocode in Figure 1.

```
Initialize the population and iteration counter
Evaluate the initial population and find the best solution
while (the stopping criterion is not met)
    Generate new solutions by modifying the existing population either locally or globally
    Evaluate the new solutions
    If the new solutions are better, then accept them and update the population
    If they are not better, accept them by some probabilistic criteria
    Update the iteration counter
end
Post-process results
```

Figure 1. Pseudocode of a typical nature-inspired algorithm.

As Figure 1 implies, all nature-inspired algorithms include both exploration and exploitation capabilities. To explore the design space more effectively, the population of candidate solutions must have sufficient diversity. On the other hand, for the algorithm to converge quickly using the fewest iterations, it must use landscape information such as gradients and modal shapes to guide its search in more promising regions. Such information exploitation can accelerate convergence. However, it can also reduce population diversity too quickly. The result is that all solutions might appear to be similar or the same and the algorithm becomes stuck.

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

Comparison with Other Algorithms

As Table 1 shows, relative to gradient-based algorithms, nature-inspired metaheuristics have distinct advantages, such as simplicity and flexibility and the ability to address hard problems. The drawback is that their computational costs tend to be higher because metaheuristics require multiple evaluations of design options and their stochastic nature makes it difficult to obtain repeat solutions.

Table 1. Advantages and disadvantages of different algorithms.

Algorithm type and categories	*Advantages*	*Disadvantages*
Gradient-based Deterministic	Higher convergence rate Efficient for local search	Gradients are not easy to estimate No guarantee for global optimality
Nature-inspired metaheuristics Stochastic Swarm intelligence	Global optimality Simple and flexible Can address hard problems Can handle diverse range of problems Easy to implement in parallel	Higher computational costs Exact solutions are not repeatable Less efficient for local search and simple problems

Algorithm Types

We selected seven examples of nature-inspired algorithms. There are many, many more, including gravitational, harmony, and wolf search algorithms and optimizations based on biogeography and the immune system.[14]

Ant colony optimization

Ants are social insects, living in organized colonies, which can have as many as 25 million ants in each colony. Ants communicate through pheromones, which are chemicals that indicate a particular ant's presence. Each ant can follow pheromone trails and lay scent pheromones to communicate with other ants. The pheromone deposition acts as positive feedback mechanism; pheromone evaporation acts as an escape mechanism from any randomly chosen trail.

The ant colony optimization algorithm, developed in 1992, exploits these local interactions and pheromone variations, essentially mimicking the ants' primary behavior and social characteristics.[8] Surprisingly, this system

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

with its simple local rules is quite effective at solving optimization problems—from vehicle routing to the well-known traveling salesman problem (TSP). The algorithm codes paths the ants visit as actual paths. The choice of which route for a particular ant at the junction of actual paths depends on the pheromone concentration on each possible path. Pheromone deposition signifies a favored route, and pheromone evaporation ensures that nonoptimal routes discovered in the earlier search stages will not converge too quickly.

Bee colony optimization

A honeybee colony typically consists of about 20,000 to 80,000 bees with one queen and a few hundred male bees or drones, with the rest being female worker bees. The workers can be scouts, onlookers, guards, or nectar collectors. A scout communicates to other workers through a waggle dance that it has found a new nectar source. An algorithm that mimics the main characteristics of the bees' foraging behavior—the artificial bee colonly algorithm—was developed in 2005..[15] The algorithm divides bees into employed, scout, and onlooker bees; codes the nectar source's location as a solution vector; and links the nectar amount with the landscape of objectives. Although this description is somewhat simplistic, it captures the main characteristics of foraging behavior. It has been used to solve unconstrained numerical optimization problems and constrained optimization problems as well as to train neural networks.

Bat algorithm

Most of the 800+ microbat species of bats use echolocation for navigation, which involves emitting about 10 to 20 ultrasonic bursts per second. Each burst lasts only a few thousandths of a second, has a frequency of 20 to 200 kHz, (humans can hear at most a 20-kHz burst), and can be as loud as 120 dB (the noise level of a jumbo jet taking off). When a bat finds an insect and is homing in on its prey, the pulse emission rate can accelerate to 200 pulses per second with a higher frequency. Echolocation allows the bats to more accurately gauge a flying insect's size, position, range, speed, and direction.

The bat algorithm, developed in 2010, uses characteristics of pulse emission and frequency tuning[16] and considers the bat's location as a solution vector in the search space. The frequency tuning lets the bat explore the search space on a larger scale, while the speedup of the pulse emission focuses on the neighborhood of local promising solutions. Among the whole bat group, there is a global best solution, and other bats tend to swarm toward it. Consequently, convergence is relatively rapid, controlled by frequency tuning, pulse emission, and loudness. The bat algorithm has been applied in many real-world applications such as engineering optimization, training neural networks, image processing, and solving the TSP.

Cuckoo search

Some cuckoo species, such as Old World cuckoos, engage in brooding parasitism, in which the cuckoo lays eggs in the nest of a host bird such as a warbler. The eggs then hatch and the host bird raises the cuckoo chicks. Because cuckoos are adept at mimicry, the texture, color, and size of the cuckoo's eggs look very similar to those of the host birds' eggs. Even so, some host birds can recognize cuckoo eggs and then get rid of them or abandon the nest, creating a kind of evolutionary arms race between the two species.

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

The cuckoo search algorithm, developed in 2009, considers a cuckoo's egg as a solution vector and the nesting field as the search space.[17] There is some evidence that both the cuckoo's and host bird's flight paths can obey Lévy flights—flights with occasional long jumps followed by many local random steps—which makes the search more effective over a large region. The similarity of eggs can be converted into the similarity of solutions, which helps the iterative search process reach convergence, and the discovery probability aids in global exploration. The cuckoo search algorithm has been successfully applied to engineering optimization and image-processing problems.

Particle swarm optimization

Birds and fish form swarms as they move, in part because there is safety in numbers. Each bird follows simple rules and often only tracks the flying status of the seven birds adjacent to it. Newtonian mechanics govern flight motion, and, amazingly, birds almost never collide. The overall swarm shows some organized structures.

Particle swarm optimization, developed in 1995, is based on these simple rules and swarming characteristics.[18] It considers a particle's position as a solution vector, so each particle has a historical best solution, and the entire swarm produces the current best solution. The simple rules are used to update each particle's position and velocity, and all particles tend to swarm toward the centroid—the global best solution. The system can converge very quickly in many cases. On the one hand, rapid convergence makes this method an effective optimizer; on the other hand, convergence might be too early, thus leading to premature convergence. Because many approaches are possible in resolving premature convergence, this algorithm has quite a few variants. Particle swarm optimization has been applied to almost every area in science and engineering, including design optimization, image process, and scheduling.

Firefly algorithm

Fireflies in tropical regions use bioluminescence to communicate, and each firefly species can have a distinct flashing pattern that serves as a unique signaling system. Other fireflies of the same species will be attracted by the flashing light and thus might swarm to it. Because light intensity decreases with distance and increased air pollution, the flashing light is visible only a few hundred meters away.

The firefly algorithm, developed in 2008, is based on these flashing characteristics. A firefly's position corresponds to a solution vector, and the objectives landscape determines its brightness or attractiveness. Because short-distance attraction is stronger than long-distance attraction, the whole firefly algorithm can be automatically subdivided into subgroups or subswarms, and each subswarm will swarm around a local model in the landscape with peaks for maximization and valleys for minimization. Consequently, the algorithm can find multiple optimal solutions simultaneously and, under certain conditions—such as the right attraction range and the monotonic decrease of its randomness—it can converge more quickly than genetic algorithms and particle swarm optimization. Like particle swarm optimization, the firefly algorithm has also been widely applied—in areas from scheduling and classification to image processing and design optimization in engineering.

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

Flower pollination algorithm

Obviously, not all algorithms are based on swarming behavior. Plants also provide ingenious problem-solving models, such as the pollination process. Of the quarter million flowering plant species, 90 percent are biotic, requiring pollen transfer through some agent, such as bats, birds, ants, and bees. Some pollinators, such as hummingbirds, tend to visit certain flower species exclusively, forming a flower constancy. In abiotic pollination—which is characteristic of the remaining 10 percent of flowering plant species—pollination tends to be local, carried out by wind or water. In contrast, biotic pollination can be global because pollinators tend to move across longer distances.

The flower pollination algorithm, developed in 2012, encodes a pollen gamete as a solution vector to a problem, uses biotic pollination to simulate global search and abiotic pollination for local search.[19] The switch between global and local search is controlled by a probability to simulate the percentage of biotic and abiotic pollination. The algorithm has been applied to solve multiobjective optimization problems.

Application Areas

Nature-inspired algorithms have been applied to a wide range of diverse applications, with literally hundreds of new articles published each year describing work to explore and solve diverse problems in real-world applications across domains.

Hard problems

A classic example of a hard problem is the TSP, which has been solved using metaheuristic approaches such as ant colony optimization.[8] The TSP is representative of how nature-inspired methods can be a powerful alternative for dealing with hard problems because existing approaches cannot solve hard problems on a realistic timescale. Although nature-inspired algorithms are not guaranteed to produce optimal solutions consistently, suboptimal solutions can still be very useful and are certainly better than no solutions at all.

Telecommunications

Nature-inspired algorithms have also been used to optimize local access networks to maximize quality of service and minimize overall energy consumption.[20] These algorithms enable better designs than those obtained by traditional methods; new design options tend to have improved efficiency with lower energy consumption and less cochannel interference.

Image processing

Image processing often concerns time-consuming computational tasks. When traditional techniques are combined with nature-inspired algorithms, features can be extracted more accurately for many applications including image segmentation, classification and deep-belief networks.[21]

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

Engineering design

Many engineering design problems are highly nonlinear—such as structural design and wireless sensor networking—and traditional methods do not handle such nonlinearity well. Recent studies show that nature-inspired algorithms can often produce better design options more efficiently because they use landscape information to search design-space regions and share information among different agents (or solution sets).[14,17,22]

Vehicle routing

Vehicle routing is a challenging problem with many applications in logistics and combinatorial optimization. Several vehicle routing problems—for example, deploying vehicles under various constraints—have been solved using nature-inspired algorithms to enhance performance. In one study, the use of metaheuristic approaches was shown to be more effective in solving vehicle routing problems with reduced transport costs.[23] The scheduling of aircraft, departure slot allocation, and airspace management can also be solved satisfactorily by nature-inspired metaheuristics, with solutions leading to reduced running costs and more effective use of departure slots.[24]

As applications become more complex and are tasked with handling big-data volume, velocity, and variety, researchers will increasingly look at how nature handles similar problems, motivating studies in important new directions. One area is theoretical analysis—understanding how nature-inspired algorithms work. Any theoretical insight gained will be extremely useful in guiding the proper use of current algorithms and in designing new algorithms that can solve challenging problems more effectively. Optimization problems remain in all areas of science, engineering, industry, and business applications, and there is little doubt that application explosion will continue.

Algorithms that move beyond mimicking nature will also be important in gaining new perspectives on problem-solving strategies. Biological systems evolve in a Darwinian manner with no high-level objective; they merely respond and adapt to environmental changes. Thus, survival of the fittest does not require global optimality as long as the fittest can survive as a population or a species. Given that nature is not always optimal, research must explore ways to give algorithms more robustness in adapting to change.

Hybrid algorithms will be of interest because they combine the best of several algorithms. How to design a better hybrid is itself a higher-level optimization problem—the optimization of optimization algorithms. Research must move beyond the trend of developing hybrids solely to claim gains in efficiency and robustness over nonhybrid algorithms. The comparative study of any hybrid algorithm should involve only hybridized counterparts.

In the Internet of Things age, algorithms must become self-adaptive and intelligent. There are far too many applications to find the best algorithm for a particular problem. In contrast, self-adaptive approaches enable the automatic choice of algorithms that can adapt for a given task set, carrying out tasks with little or no user intervention. Self-adaptation also means being able to control performance and self-tune parameters to

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

maintain the highest efficiency. The ultimate goal is to develop an intelligent toolbox that can solve the problem at hand with no more user effort than pressing a button.

Obviously, these are only a few possible directions. A combination of new algorithms coupled with traditional techniques can be very useful to pursue. After all, traditional techniques have been well established and widely tested, and they are among the most useful to a specific class of problems. New methods will be most needed when traditional methods do not work well. However, given application demands, we expect to see more research in nature-inspired algorithms to evolve more efficient tools for solving the diversity of problems in real-world applications.

References

1. X.S. Yang, *Engineering Optimization: An Introduction with Metaheuristic Applications*, John Wiley and Sons, 2010.
2. L. Fisher, *The Perfect Swarm: The Science of Complexity in Everday Life*, Basic Books, 2009.
3. P. Miller, "Swarm Theory," *National Geographic*, July 2007.
4. J. Surowiecki, *The Wisdom of Crowds*, Anchor Books, 2004.
5. W.R. Ashby, "Principles of the Self-Organizing System," *Trans. Symp. Univ. of Illinois: Principles of Self-Organization*, H. Von Foerster and G.W. Zopf, Jr., eds., Pergamon Press, 1962, pp. 255–278.
6. E.F. Keller, "Organisms, Machines, and Thunderstorms: A History of Self-Organization, Part Two: Complexity, Emergence, and Stable Attractors," *Historical Studies in the Natural Sciences*, vol. 39, no. 1, 2009, pp. 1–31.
7. X.S. Yang, *Nature-Inspired Optimization Algorithms*, Elsevier, 2014.
8. E. Bonabeau, M. Dorigo, and G. Theraulaz, S*warm Intelligence: From Natural to Artificial Systems*, Oxford Univ. Press, 1999.
9. S. Kirpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, no. 4598, 1983, pp. 671–680.
10. C. Blum and A. Roli, Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison, *ACM Computing Surveys*, vol. 35, no. 2, 2003, pp. 268–308.
11. A. Gut, *Probability: A Graduate Course*, Springer, 2005.
12. G. Marsaglia and A. Zaman, "Monkey Tests for Random Number Generators," *Computers & Mathematics with Applications*, vol. 26, no. 9, 1993, pp. 1–10.
13. G. Lorge, "The Best Thought Experiments: Schrödinger's Cat, Borel's Monkeys," *Wired*, 22 May 2007; www.wired.com/2007/05/st-best-4.
14. X.S. Yang et al., *Swarm Intelligence and Bio-inspired Computation: Theory and Applications*, Elsevier, 2013.
15. D. Karaboga, *An Idea Based on Honey Bee Swarm for Numerical Optimization*, Erciyes Univ., Computer Eng. Dept., Tech. Report TR06, 2005.
16. X.S. Yang, "A New Metaheuristic Bat-Inspired Algorithm," *Proc. Conf. Nature-Inspired Cooperative Strategies for Optimization* (NICSO 10), J.R. Gonzales, ed., *Studies in Computational Intelligence,* vol. 284, 2010, pp. 65–74.
17. X.S. Yang and S. Deb, "Multiobjective Cuckoo Search for Design Optimization," *Computers and Operations Research*, vol. 40, no. 6, 2013, pp. 1616–1624.
18. J. Kennedy and R.C. Eberhart, "Particle Swarm Optimization," *Proc. IEEE Int'l Conf. Neural Networks* (ICNN 95), 1995, pp. 1942–1948.

IEEE Computer (Magazine), vol. 49, no. 9, pp. 52-59 (2016).

DOI: 10.1109/MC.2016.292 http://ieeexplore.ieee.org/document/7562319/

19. X.S. Yang, M. Karamanoglu, and X.S. He, "Flower Pollination Algorithm: A Novel Approach for Multiobjective Optimization," *Engineering Optimization*, vol. 46, no. 9, 2014, pp. 1222–1237.
20. X.S. Yang, S.F. Chien, and T O. Ting, *Bio-inspired Computation in Telecommunications*, Elsevier, 2015.
21. X.S. Yang and J.P. Papa, *Bio-inspired Computation and Applications in Image Processing*, Elsevier, 2016.
22. H.E.P. Espinosa, *Nature-Inspired Computing for Control Systems*, Springer, 2016.
23. N. Labadie, C. Prins, and C. Prodhon, *Metaheuristics for Vehicle Routing Problems*, John Wiley and Sons, 2016.
24. N. Durand et al., *Metaheuristics for Air Traffic Management*, John Wiley and Sons, 2016.

Elements of an Optimization Problem

A typical optimization problem has a main design objective such as cost or energy efficiency, but is usually subject to many design constraints, which necessitates two optimization stages: problem formulation to define and prioritize constraints and solution search to solve the problem with the optimal method.

Problem Formulation

To illustrate problem formulation in optimization, consider the hunt for a missing black box from a trans-Atlantic flight. The search's design constraints—budget, ocean currents, weather, and time—will be contradictory. Weather can cause delays, which might increase cost and work against the goal of finding the box within a certain time. There is also a stringent time constraint because the box's battery has a relatively short life. Thus, problem formulation must consider certain tradeoffs among cost, time, and the probability of finding the box, and these tradeoffs are subject to changes in various other factors, such as resources and location accessibility.

Solution Search

Once an optimization problem is properly formulated, the main task is to find its optimal solution. Information about the box's possible location might come from the flight's last known position, and the search might start in that region. However, currents could have carried the box to another region, making the initial search quasi-random. A single agent hunting every possible square inch of the ocean region would be exhaustive but hardly practical. An alternative is to have a group search, in which members share information. The shared knowledge leads to a collective intelligence that helps narrow the search space.

In reality, most search processes are somewhere between random and focused; searching is not purely random even when there are no clues about where to look. As in the black-box example, the most likely scenario is a quasi-random search that seeks hints. It starts with a plausible location and then moves to another plausible location in the neighborhood and so on. Hints are exchanged with others in the search group. Searchers who are industrious and apply their experience to uncover more possible solutions might be rewarded or otherwise encouraged. Likewise, lazy or less motivated searchers might be dismissed. In this way, the selection of searchers and solutions will gradually improve the probability of finding the black box. This process of refined collective intelligence leading to a solution is the essence of contemporary metaheuristic optimization algorithms based on swarm intelligence.